UNITED STATES PATENT AND TRADEMARK OFFICE

MN

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/821,085	04/08/2004	Cindy Howard	SBS001/150068	1137

23444 7590 06/21/2007

ANDREWS & KURTH, L.L.P.
600 TRAVIS, SUITE 4200
HOUSTON, TX 77002

EXAMINER	
WANG, JUE S	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
06/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/821,085	HOWARD ET AL.	
	Examiner	Art Unit	
	Jue S. Wang	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 April 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *08 April 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *08 April 2004*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-27 have been examined.

Claim Objections

2. Claims 1-7 and 23 are objected to because of the following informalities:

Claim 1, the punctuation mark following "comprising the steps of" in line 2 should be ":" instead of ",".

Claims 2-6, the punctuation mark following "further comprising the steps of" in line 1 should be ":" instead of ",".

Claim 7, the punctuation mark following "comprising" in line 2 should be ":" instead of ",".

Claim 23, the punctuation mark following "comprising" in line 2 should be ":" instead of ",".

Appropriate corrections are required.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 1-22 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

A. The following claim language is not clear and indefinite:

i. As per claim 1, lines 5-6, the phrase "storing said elements in a database which retains said interrelationships between said elements". This limitation is not clearly understood (i.e., are the interrelationships between the elements and the elements two separate entities that are both stored in the database, or do the elements themselves retain the interrelationships and only the elements are stored in the database, etc.).

ii. As per claim 7, lines 1-2 and 9, the term "business logic" is used. This limitation is not clearly understood because "business logic" is a non-standard terminology that has varying definitions (i.e., the part of an application program that performs the required data processing of the business, or the business rules that express business policy, etc.). Furthermore, in claim 7, line 10, the phrase "said template characterized by" defines the template indefinitely. Claims have to be defined definitely and clearly.

Appropriate corrections are required.

Any claim not specifically addressed, above, is being rejected as incorporating the deficiencies of a claim upon which it depends.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1, 7, 9, 10, 13-15, and 23 are rejected under 35 U.S.C. 102(b) as being anticipated by Fuchs et al. (US 2002/0026632 A1, hereinafter Fuchs).

7. As per claim 1, Fuchs teaches the invention as claimed including a method for converting original computer source code to target source code (see [0001]) comprising the steps of:

resolving said original computer source code into a plurality of basic constituent elements and a plurality of interrelationships therebetween (see Fig 2, [0009], [0016], and [0017]),

storing said elements in a database which retains said interrelationships between said elements (see [0020]), and

writing said target source code from at least one template incorporating said elements therein, said template defining the structure of said target source code and comprising an algorithmic template language for controlling said writing (see Fig 1, [0018], [0021]-[0023]).

8. As per claim 7, Fuchs teaches the invention as claimed including a system for converting original computer source code representing a business logic to target source code comprising (see [0001]):

at least one computer system (EN: the code generator must be executed by a computer system since the code generation is automatic, see [0001] and [0002]) having at least one template (see Fig 1, item 3, [0010], and [0018]), a deconstruction program (see Fig 1, [0010], [0016]-[0017]), a database operably coupled with said deconstruction program, and a construction program (see Fig 1 and [0018]) operably coupled with said database (EN: Fuchs does not explicitly teach that the system has a database to which the deconstruction program and

the construction programs are coupled, however, Fuchs teaches that the intermediate file generated by the front end may be stored and accessed by the template which is used in the back end to generate the target code (see [0017], [0020], and [0022]). Therefore, the intermediate file must be stored in a data storage of some sort (i.e., database) and that the front end and back end are coupled to this data storage since the front end stores the file and the back end access the file.)

said original source code forming an input to said deconstruction program resulting in an output of said deconstruction program of a plurality of basic elements and a plurality of interrelationships therebetween, said output of said deconstruction program retaining said business logic and stored in said database (see Fig 2, [0009], [0016], and [0017]),

said template characterized by an algorithmic template language included therein which controls said construction program and which generally defines the structure of said target source code (see Fig 1, [0018], [0021]-[0023]),

said construction program arranged and designed to receive said plurality of basic elements and said plurality of interrelationships therebetween stored in said database as a first input, receive said at least one template as a second input, and produce as an output said target source code as an output (see Fig 1 and [0018]).

9. As per claim 9, Fuchs further teaches that the deconstruction program includes at least one language-dependent parser (see Fig 1 and [0016]; EN: grammatical analysis is done by a parser and the parser is language dependent as it uses a different grammar to parse the file depending on the language).

10. As per claim 10, Fuchs further teaches said database stores said original code (see [0002]; EN: the universal code generator provides for automatic code generation, so the original source must be stored in a data storage of some sort since it must be access by the front end during the automatic code generation process. Furthermore, the data storage is the same one as the data storage for storing the intermediate file since regardless of whether the original source code is initially stored on the same system as the code generator or remotely on a networked storage, it must be stored locally first before the front end can access the original source code and likewise, the intermediate file must be stored locally to facilitate access by the back end).

11. As per claim 13, Fuchs further teaches that said database stores at least one rule for controlling said construction program (see [0002] and [0021]; EN: the universal code generator provides for automatic code generation, so the template which contains rules for controlling the back end must be stored in a data storage of some sort since it must be access by the back end during the automatic code generation process).

12. As per claim 14, Fuchs further teaches that said database stores at least one core construct (see Fig 2, [0002], and [0019]; EN: the intermediate file containing the syntactic tree is stored where the nodes of the syntactic tree corresponds to core programming language constructs such as modules, interfaces, variables, and constants).

13. As per claim 15, Fuchs further teaches that said database stores at least one template (see [0002]; EN: that the universal code generator provides for automatic code generation, so the

template must be stored in a data storage of some sort since it must be access by the back end during the automatic code generation process).

14. As per claim 23, Fuchs teaches the invention as claimed including a system for converting original computer source code to target source code comprising:

at least one computer system having a deconstruction means (see Fig 1, [0010], [0016]-[0017]), a database (see [0017] and [0020]), at least one template (see Fig 1, item 3, [0010], and [0018]), and a construction means (see Fig 1 and [0018]),

said deconstruction means being arranged and designed to resolve said original source code into a plurality of basic constituent elements and a plurality of interrelationships therebetween (see Fig 2, [0009], [0016], and [0017]),

said database adapted for storing said elements and interrelationships (see [0017] and [0020]),

said template defining generally the structure of said target source code and comprising an algorithmic template language designed and arranged for controlling said construction means (see Fig 1, [0018], [0021]-[0023]), wherein

said construction means is arranged and designed to interpret said template and write said target source code therefrom, incorporating said elements and interrelationships as directed by said template (see Fig 1 and [0018]).

Claim Rejections - 35 USC § 103

15. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

16. Claims 2-6, 18-22, and 24-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fuchs et al. (US 2002/0026632 A1, hereinafter Fuchs), in view of Uner (WO 02/35349 A1).

17. As per claim 2, Fuchs does not teach including a condition statement in said at least one template, said condition statement having a condition expression contained therein, evaluating said condition expression, performing a first task if said condition expression is evaluated to be a first value, and performing a second task if said condition expression is evaluated to be a second value.

Uner teaches a method for translating data streams by using instructions and associated data stored in templates, where the first template containing the instructions is in a high level programming language which includes Genesis Template Markup Language, C, C+, C++, PASCAL, FORTRAN, COBOL, or the like, the Genesis Template Markup Language features a rich set of instructions and is a complete programming language with conditional statements (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly disclose that the template contains a conditional statement, it is obvious that the template can contain a conditional

statement since the template language used to describe the template provides a means for constructing conditional statements.

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying conditional statements so that templates can contain conditional statements as taught by Uner because it is reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

18. As per claim 3, Fuchs does not teach including an iteration statement in said at least one template, said iteration statement having an iteration expression contained therein, iteratively evaluating said iteration expression, and performing a task while said iteration expression is evaluated to be a particular value.

Uner teaches that the template is written in a high level programming language which contains an iteration statement (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly disclose that the template contains an iteration statement, it is obvious that the template can contain an iteration statement since the template language used to describe the template provides a means for constructing iteration statements.

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying iteration statements so that templates can contain iteration statements as taught by Uner because it is

reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

19. As per claim 4, Fuchs does not teach including a subroutine statement in said template, said subroutine statement having a subroutine name contained therein, designating a portion of said at least one template by said subroutine name, and interpreting said subroutine statement as a directive to control said-writing by said portion.

Uner teaches that the template is written in a high level programming language, (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly state as such, it is well known in the art that high level programming languages provide a means to construct subroutines. While Uner does not explicitly disclose that the template contains a subroutine statement, it is obvious that the template can contain a subroutine statement since the template language used to describe the template provides a means for constructing subroutine statements.

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying subroutine statements so that templates can contain subroutine statements as taught by Uner because it is reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

20. As per claim 5, Fuchs does not teach when said portion has completed controlling said writing, controlling said writing by said template language immediately following said subroutine statement.

Uner teaches that the template is written in a high level programming language, (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly state that the high level programming language provides subroutine statements, it is well known in the art that high level programming languages provide a means to construct subroutines. Furthermore, it is well known in the art that that when subroutines finish execution, the flow of the program execution returns to the line of code immediate after the call of the subroutine.

21. As per claim 6, Fuchs does not teach including a set statement in said at least one template, said set statement having a set expression and a set variable contained therein, assigning said set expression to said set variable.

Uner teaches that the template is written in a high level programming language, see page 13, lines 1-6 and lines 12-16. While Uner does not explicitly state as such, it is well know in the art that any high level programming language would contain expressions for variable assignment. While Uner does not explicitly disclose that the template contains a set expression, it is obvious that the template can contain a set expression since the template language used to describe the template provides a means for constructing set expression.

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying set expressions so that templates can contain set expressions as taught by Uner because it is

reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

22. As per claim 18, Fuchs does not teach said template language includes a condition statement having a condition expression contained therein, said construction program being designed and arranged to interpret said condition statement as a directive to evaluate said condition expression and perform a first task if said condition expression is evaluated to be a first value and perform a second task if said condition expression is evaluated to be a second value.

Uner teaches the template is written in a high level programming language which contains a conditional statement (see page 13, lines 1-6 and lines 12-16).

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying conditional statements as taught by Uner because it is reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

23. As per claim 19, Fuchs does not teach said template language includes an iteration statement having an iteration expression contained therein, said construction program being

designed and arranged to interpret said iteration statement as a directive to iteratively evaluate said iteration expression and perform a task while said iteration expression is evaluated to be a particular value.

Uner teaches the template is written in a high level programming language which contains an iteration statement (see page 13, lines 1-6 and lines 12-16).

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying looping statements as taught by Uner because it is reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

24. As per claim 20, Fuchs does not teach said template language includes a subroutine statement having a subroutine name contained therein, said at least one template having a portion therein designated by said subroutine name, said construction program being designed and arranged to interpret said subroutine statement as a directive to interpret said portion.

Uner teaches that the template is written in a high level programming language (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly state as such, it is well known in the art that high level programming languages provide a means to construct subroutines.

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying

subroutine statements as taught by Uner because it is reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements, see [0023] of Fuchs, so the template language of Fuchs must be at least as expressive as the template language of Uner.

25. As per claim 21, Fuchs does not teach that when said construction program has completed interpreting said portion, said construction program interprets said template language immediately following said subroutine statement.

Uner teaches that the template is written in a high level programming language (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly state as such, it is well known in the art that high level programming languages provide a means to construct subroutines. Furthermore, it is well known in the art that that when subroutines finish execution, the flow of the program execution returns to the line of code immediate after the call of the subroutine.

26. As per claim 22, Fuchs does not teach said template language said includes a set statement having a set expression and a set variable contained therein, said construction program being designed and arranged to interpret said set statement as a directive to assign said set expression to said set variable.

Uner teaches that the template is written in a high level programming language (see page 13, lines 1-6 and lines 12-16). While Uner does not explicitly state as such, it is well know in the

art that any high level programming language would contain expressions for variable assignment.

It would have been obvious to one of ordinary skill in the art at the time of the invention that the Template Description Language (TDL) of Fuchs contains a means of specifying set statements as taught by Uner because it is reasonable to expect that the complexity of translating code from one language to another is greater than translating a data stream as TDL may be a complete programming language that deals with conventional software elements (see [0023] of Fuchs), so the template language of Fuchs must be at least as expressive as the template language of Uner.

27. As per claims 24-27, they recite features of the template language that are also recited in claims 18-20 and 22. Therefore, they are rejected using the same reasons as claims 18-20 and 22.

28. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Fuchs et al. (US 2002/0026632 A1, hereinafter Fuchs), in view of Venter, (US 7,219,338 B2,).

29. As per claim 8, Fuchs does not teach that the deconstruction program includes a language-determination parser.

Venter teaches a method for multi-language compilation (see abstract), including a language-determination parser (see Fig 2 and column 5, lines 62-65).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Fuchs to contain a language-determination parser as taught by Venter because

it facilitates a faster parsing process. In Venter, the parsers are predefined for each specific language and only a determination of which parser to use is required, whereas in Fuchs, the parser must be configured to handle the particular language being parsed by using the grammar as an input (see [0016] of Fuchs), so the overhead associated with the configuration process is avoided in Venter by using predefined parsers.

30. Claims 11 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fuchs et al. (US 2002/0026632 A1, hereinafter Fuchs), in view of Rechter, (US 6,698,014 B1).

31. As per claim 11, Fuchs does not teach said database stores an original environment description.

Rechter teaches a system for automatically converting source code from one programming language to another, including storing in a database an original environment description (see column 3, lines 35-38, column 4, lines 50-59, column 6, lines 49-59).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Fuchs such that an original environment description is stored in the database as taught by Rechter because it is necessary to know the supplier, version, and release of the operating system, development environments, databases, and software language of the source and target environments to convert source code from a first programming language and platform to a second programming language and platform (see column 2, lines 25-38 and column 50-59 of Rechter).

32. As per claim 12, Fuchs does not teach said database stores a target environment description.

Rechter teaches a system for automatically converting source code from one programming language to another, including storing in a database a target environment description (see column 3, lines 35-38, column 4, lines 50-59, column 6, lines 49-59).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Fuchs such that a target environment description is stored in the database as taught by Rechter because it is necessary to know the supplier, version, and release of the operating system, development environments, databases, and software language of the source and target environments to convert source code from a first programming language and platform to a second programming language and platform (see column 2, lines 25-38 and column 50-59 of Rechter).

33. Claims 16 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fuchs et al. (US 2002/0026632 A1, hereinafter Fuchs), in view of Ono, (US 6,219,831 B1).

34. As per claim 16, Fuchs does not teach said construction program includes a template workbench designed and arrange for editing said at least one template.

Ono teaches a device for converting computer programming languages (see abstract), including a template workbench designed and arrange for editing said at least one template (see Fig 3, column 3, lines 9-21, and column 5, lines 41-59; EN: templates consist of rules, so the

conversion rule input screen which is an interface to edit rules within the template is considered as an interface to edit the template).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Fuchs to provide a template workbench designed and arrange for editing the at least one template as taught by Ono because it facilitates maintenance of templates (see column 5, lines 46-63 of Ono).

35. As per claim 17, Fuchs does not teach said construction program includes a rules workbench designed and arranged for editing said at least one rule.

Ono teaches a rules workbench designed and arranged for editing said at least one rule (see Fig 3, column 3, lines 9-21, and column 5, lines 41-59).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Fuchs to provide a rules workbench designed and arrange for editing the at least one rule as taught by Ono because it facilitates maintenance of conversion rules (see column 5, lines 46-63 of Ono).

Conclusion

36. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Andrews et al. (US 5,768,564) is cited to teach a method and apparatus for translating source code from one high-level computer language to another.
- Gelfenbain (US 5,898,874) is cited to teach dynamic codeset translation environment.

- Moser et al. (US 6,275,789 B1) is cited to teach method and apparatus for performing full bidirectional translation between a source language and a linked alternative language.
- Simser (US 6,314,429 B1) is cited to teach a bi-directional conversion library for translating data structures used in a computer program from a first language to data structures used by a second computer programming language.
- Sullivan (US 6,453,464 B1) is cited to teach a method and apparatus for converting Cobol to Java.
- Hughes et al. (US 6,519,768 B1) is cited to teach instruction translation method.
- Dupuy et al. (US 6,523,171 B1) is cited to teach enhanced source code translator from procedural programming language to an object oriented programming language.
- Jayaram et al. (US 6,996,589 B1) is cited to teach a system and method for database conversion.
- Tondreau et al. (US 2003/0226132 A1) is cited to teach a method and system for transforming legacy software applications into modern object oriented systems.
- McNamara (US 2003/0200535 A1) is cited to teach a system for program source code conversion.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jue S. Wang whose telephone number is (571) 270-1655. The examiner can normally be reached on M-Th 7:300 am - 5:00pm (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 571-272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

J.W.
5/25/2007

MENG-AI T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100